Unsupervised Extra Trees: a stochastic approach to compute similarities in heterogeneous data.

Kevin Dalleau, Miguel Couceiro, Malika Smaïl-Tabbone

Noname manuscript No.
(will be inserted by the editor)

Unsupervised Extra Trees: a stochastic approach to compute similarities in heterogeneous data.

Kevin Dalleau · Miguel Couceiro · Malika Smail-Tabbone

the date of receipt and acceptance should be inserted later

Abstract In this paper we present a method to compute similarities on unlabeled data, based on extremely randomized trees. The main idea of our method, Unsupervised Extremely Randomized Trees (UET) is to randomly split the data in an iterative fashion until a stopping criterion is met, and to compute a similarity based on the co-occurrence of samples in the leaves of each generated tree. Using a tree-based approach to compute similarities is interesting, as the inherent We evaluate our method on synthetic and real-world datasets by comparing the mean similarities between samples with the same label and the mean similarities between samples with different labels. These metrics are similar to intracluster and intercluster similarities, and are used to assess the computed similarities instead of a clustering algorithm's results. Our empirical study shows that the method effectively gives distinct similarity values between samples belonging to different clusters, and gives indiscernible values when there is no cluster structure. We also assess some interesting properties such as invariance under monotone transformations of variables and robustness to correlated variables and noise. Finally, we performed hierarchical agglomerative clustering on synthetic and real-world homogeneous and heterogeneous datasets using UET versus standard similarity measures. Our experiments show that the algorithm outperforms existing methods in some cases, and can reduce the amount of preprocessing needed with many real-world datasets.

Keywords Similarity measure · clustering · unsupervised classification · decision tree · extremely randomized trees

This paper is an extended version of the PAKDD'2018 Long Presentation paper "Unsupervised Extremely Randomized Trees".

Universite de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
E-mail: Kevin Dalleau
kevin.dalleau@loria.fr

Miguel Couceiro
miguel.couceiro@loria.fr

Malika Smal-Tabbone
malika.smail@loria.fr

1 Introduction and preliminaries

Many unsupervised learning algorithms rely on a metric to evaluate the pairwise distance between samples. Despite the large number of metrics already described in the literature [1], in many applications, the set of available metrics is reduced by intrinsic characteristics of the data and of the chosen algorithm. The choice of a metric may strongly impact the quality of the resulting clustering, thus making this choice rather critical.

Shi and Horvath [2] proposed a method to compute distances between instances in unsupervised settings using Random Forest (RF). RF [3] is a popular algorithm for supervised learning tasks, and has been used in many fields, such as biology [4] and image recognition [5]. It is an ensemble method, combining decision trees in order to obtain better results in supervised learning tasks with highdimensional data. Let $L = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a training set, where $X = \{x_1, \ldots, x_n\}$ is a list of *samples* (*i.e.*, feature vectors) and $Y = \{y_1, \ldots, y_n\}$ is the list of corresponding class labels. The algorithm begins by creating several new training sets, each one being a bootstrap sample of elements from X. A decision tree is built on each training set, using a random sample of m_{try} features at each split. The prediction task is then performed by a majority vote or by averaging the results of each decision

tree, according to the problem at hand (classification or regression). This approach leads to better accuracy and generalization capacity of the model compared to single decision trees, while reducing the variance [6].

The method proposed by Shi and Horvath in [2], called unsupervised random forest (URF), derives from the common RF algorithm. Once the forest has been constructed, the training data can be run down each tree. Since each leaf only contains a small number of instances, and all objects of the same leaf can be considered similar, it is possible to define a similarity measure from these trees: if two objects i and j are in the same leaf of a tree, the overall similarity between the two objects is increased by one. This similarity is then normalized by dividing by the number of trees in the forest. In doing so, the similarities lie in the interval $[0, 1]$. The use of RF is made possible in the unsupervised case thanks to the generation of synthetic instances, enabling binary classification between the latter and the observed, unlabeled instances. Two methods for synthetic data generation are presented in [2], namely, *addCl1* and *addCl2*.

In *addCl1*, the synthetic instances are obtained by a random sampling from the observed distributions of variables, whereas in *addCl2* they are obtained by a random sampling in the hyper-rectangle containing the observed instances. The authors found that *addCl1* usually leads to better results in practice. URF as a method for measuring dissimilarity presents several advantages. For instance, objects described by mixed types of variables as well as missing values can be handled. The method has been successfully applied in fields such as biology [7–9] and image processing [10].

Albeit its appealing character, the method suffers from two main drawbacks. Firstly, the generation step is not computationally efficient: since the obtained trees highly depend on the generated instances, it is necessary to construct many forests with different synthetic instances and average their results, leading to a computational burden. Secondly, the synthetic instances may bias the model being constructed to discriminate objects on specific features. For example, *addCl1* leads to forests that focus on correlated features.

P.Geurts, D.Ernst and L. Wehenkel [11] presented a novel type of tree-based ensemble method that they called Extremely Randomized Trees (or ExtraTrees, for short). This algorithm is very similar to RF. In RF, both instances and features are sampled during the construction of each tree. In ExtraTrees (ET) another layer of randomization is added: whereas in RF the threshold of a feature split is selected according to some purity measure (the most popular ones being the entropy and the Gini impurity), in ET these thresholds are obtained totally or partially at random. Moreover, instead of growing the trees from bootstrapped samples of the data, ET uses the whole training set. At each node, K attributes are randomly selected and a random split is performed on each one of them. The best split is kept and used to grow the tree.

Two parameters are of importance in ET: K, defined above, and n_{min}, that is the minimum sample size for a node to be split. Interestingly, the parameter K, that takes values in $\{1, ..., n_{features}\}$, influences the randomness of the trees. Indeed, for small values of K, the dependence of the constructed trees on the class labels gets weak. In the extreme case where K is set to 1 (*i.e.*, only one feature is selected and randomly split), the dependence of the trees on the observed label is removed. Following the tracks of [2] on URF, we propose to use ET with a novel approach where the synthetic case generation is no longer necessary. We call it unsupervised extremely randomized trees (UET). We also extend this method to make it applicable on heterogeneous data. Handling this type of data is a frequent issue in practice, and constitutes an active area of research [12,13].

This paper is organized as follows. We begin by presenting UET, and discuss how it can be used to compute similarities between samples in unsupervised settings. Moreover, we show how it can be applied to heterogeneous data while avoiding both pre- and post-processing phases. We then discuss some important parameters of UET and their optimization. It is noticeable that the algorithm is essentially sensitive to one parameter, the minimum number of samples for a node to split, or *smoothing strength*. We then study the behavior of UET according to different aspects in Section 3. We first validate its ability to discriminate samples from different clusters while keeping similarities constant when there is no cluster structure. We then verify empirically its robustness with respect to monotone transformations, to variable correlations and to noise (Subsection 3.1.1). Our experiments show that UET is indeed robust to these common data alterations, which is an interesting feature in practice. In Subsection 3.2 we evaluate the performance of UET on purely numerical, purely categorical, and on synthetic heterogeneous datasets. Moreover, we apply UET to some benchmark datasets, which reveals some outperforming results compared to those in the literature (Subsection 3.3). Finally, we present a general discussion of results and hint at some directions of current and future work.

2 Unsupervised Extremely Randomized Trees

2.1 Description of UET

In URF, two methods are used for the generation of synthetic data: *addCl1* and *addCl2*. Both methods work by performing a random sampling in the observed data. The synthetic data is assigned a label, while the observed data is assigned another one, enabling binary classification between observed and synthetic examples. Instead of generating new instances, another approach is to randomly assign labels to the observed data. This method, that we propose and refer to as *addCl3*, can be implemented as follows:

1. Let n_{obs} be the number of instances in the dataset. A list containing $\lfloor \frac{n_{obs}}{2} \rfloor$ times the label 0 and $n_{obs} - \lfloor \frac{n_{obs}}{2} \rfloor$ times the label 1 is generated.
2. For each instance in the dataset, a label is randomly sampled without replacement from the aforementioned list.

This procedure ensures that the label distribution is balanced in the dataset. However, this leads to the same problem arising with *addCl1* and *addCl2*: the results are highly dependent on the generation step, as different realizations of the instance-label association or of the synthetic data may lead to completely different forests. To circumvent this issue, one solution is to run multiple forests on multiple generated datasets, and to average the results. In [2], the authors found that averaging the results from 5 forests, with a total of 5000 trees leads to robust results. Moreover, instead of running multiple forests on many generated datasets, it is possible - and computationally more efficient - to run a single forest with a large amount of generated data, if some care is taken regarding the reweighting of each class. This workaround was proposed by a reviewer of [2], and is easier to implement when *addCl3* is used. Indeed, since no new instances are added, it is not necessary to reweight each class.

With *addCl3*, the construction of the trees no longer depends on the structure of the data: when *addCl1* or *addCl2* are used, the forests are trained to distinguish between observed and synthetic instances. In *addCl3*, the labels being assigned randomly, two similar instances may be labeled differently and may fall in different leaves. However, using Extra Trees (ET) with the number of features randomly selected at each node $K = 1$, the construction of the trees no longer depends on the class label, as described previously section. Hence, ET seems to be a suitable algorithm to use with *addCl3*. Algorithm 1 describes UET. The algorithm is split into two steps: (i) the ensemble of trees are constructed, and (ii) the similarity is computed using their leaves.

Algorithm 1: Unsupervised Extremely Randomized Trees

```
Data: Observations O
Result: Similarity matrix S
D ⟵ addCl3(O);
T ⟵ Build_an_extra_tree_ensemble(D, K)   // Here K = 1;
S = 0_{n_obs, n_obs}   // Initialization of a zero matrix of size n_obs ;
for d_i ∈ D do
    for d_j ∈ D do
        S_{i,j} = number of times the samples d_i and d_j fall in the same leaf node in each tree of T = {t_1, t_2, ..., t_M};
    end
end
S_{i,j} = S_{i,j} / M;
```

Let N be the number of instance of a dataset D. Computational complexity of Build_an_extra_tree_ensemble (D,K) is on the order of $Nlog(N)$ [11]. As the similarity matrix computation complexity is in $O(N^2)$, the overall complexity of UET is $O(N^2)$. One of the interesting features of this approach is the fact that a split can be agnostic to the variable type, with a splitting procedure defined accordingly. Indeed, for a continuous variable with values in a finite set $A = \{a_1, a_2, \ldots, a_n\}$, the random split can be done by sampling a cut-point in $\mathcal{U}(min(A), max(A))$, where $\mathcal{U}(a, b)$ is the continuous uniform distribution with a and b as boundaries. Although this splitting procedure makes sense in continuous and ordinal settings, as there is a notion of order between values of A, for purely categorical data there is no such ordering between modalities. A good example is given by L. Kaufman and P.J. Rousseeuw [14] :

[...] we could choose 1 = blue eyes, 2 = brown eyes, 3 = green eyes, and 4 = gray eyes. [...] it is not because gray eyes are given a higher code number than brown eyes that they would in some sense be better.

The process of sampling a cut-point in the categorical setting can be attained by randomly sampling a modality out of all the possible modalities. A binary split is then performed based on this value, *i.e.* all samples having this modality for this variable end up in a first node, while the other samples are grouped in a second node.

The procedure Build_an_extra_tree_ensemble(D,K) is essentially the one presented by Geurts *et al.* in [11], with a different splitting method, that we present in Algorithm 2.

Algorithm 2: Random split procedure

Data: Values A, a type t
Result: a split s
if $t ==$ *continuous or* $t ==$ *ordinal* **then**
  cut-point a_c drawn from $\mathcal{U}(min(A), min(B))$;
  return the split $[a < a_c]$ and $[a \geq a_c]$;
end
if $t ==$ *categorical* **then**
  cut-point a_c drawn from set(A);
  return the split $[a \in set(A) \backslash a_c]$ and $[a = a_c]$;
end

Notice that a random label generation procedure such as *addCl3* is not even necessary. Indeed, as the generated labels do not carry any information about the instances and K is set to 1, each split is performed without any consideration of the label. This enables further reduction of the algorithm's overhead. Two parameters can influence the results of the method we propose, UET:

1. The number of trees n_{trees}.
2. The minimum number of samples for a node to be split n_{min}.

As the influence of these parameters on the similarities computed by UET is not known, it is necessary to evaluate the behaviour of the method when they vary.

2.2 Optimization of parameters

For each evaluation presented in this subsection, the following process is repeated 10 times:

1. A similarity matrix is constructed using UET.
2. This similarity matrix is transformed into a distance matrix using the relation $DIS_{ij} = \sqrt{1 - SIM_{ij}}$, as used in [2].
3. An agglomerative clustering (with average linkage) is performed using this distance matrix, with the relevant number of clusters for the dataset.

For each clustering, Adjusted Rand Index (ARI) is computed. The ARI quantifies the agreement between two partitions of a dataset [15,16]. Let Table 1 be the contingency table of two clustering results, where each value n_{ij} is the number of instances associated with cluster i of Y and cluster j of X. Given this table, the ARI is defined by:

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{n}{2}} \quad (1)$$

ARI values of 1 indicates perfect agreement up to a permutation, while a value of 0 indicates a result no better than a random label assignment. Here, the ARI values presented are multiplied by 100.

	Y_1	$\cdots$	Y_q	*sums*
X_1	n_{11}	$\cdots$	n_{1q}	a_1
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_r	n_{r1}	$\cdots$	n_{qr}	a_r
sums	b_1	$\cdots$	b_q	

Table 1 Contingency table of two clusterings

Dataset	# samples	# features	# labels
Iris	150	4	3
Wine	178	13	3
Wisconsin	699	9	2

Table 2 Properties of used datasets

Three datasets are used for this evaluation process: Iris [17], Wine [18] and Wisconsin [19]. These datasets are described Table 2.2.

2.2.1 Influence of the number of trees

The influence of n_{trees} has also been studied in [11], where this parameter is referred to as the averaging strength M. For stochastic methods such as RF and ET used in a supervised learning setting, the average error is a monotonically decreasing function of M [3]. We assessed the influence of this parameter on UET results, using the protocol described above. Our experiments show that there is no substantial gain for $n_{trees} > 50$.

We compared the ARI using the Kruskal-Wallis test [20]. This non-parametric methods enables us to test whether there is a statistically significant difference between the obtained ARI. The results show that the difference in ARI in Wine, Iris and Wisconsin datasets are not significant ($p > 0.1$). This observation confirms the one from Geurts *et al.* where values of $n_{trees} > 40$ outperforms Tree Bagging. However, as the time to construct the tree ensemble grows linearly with the number of trees, it is a good option to choose small a value of n_{trees}. We chose the value $n_{trees} = 200$ by default. This value is way below the overall number of trees recommended for URF, 5000. However, the best value for n_{trees} may depend on the size of the dataset. The results are presented Figure 1.

2.2.2 Influence of the minimum number of samples to split

ET tends to produce trees having 3 to 4 times the number of leaves than those of RF. As UET computes similarities by counting the number of times objects fall into the same leaf, the results are impacted by this increase in the number of leaves. It may be useful to stop the

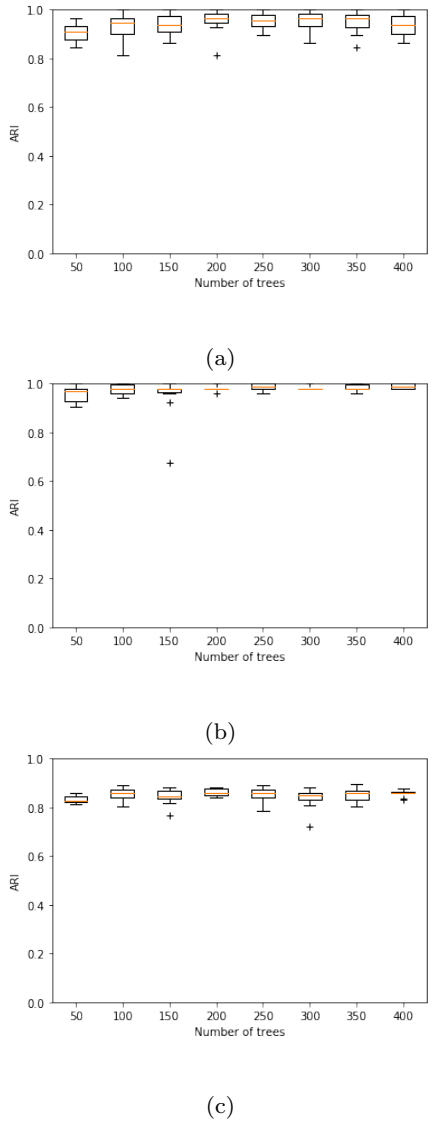

Fig. 1 ARI performing UET and agglomerative clustering on Wine (a), Iris (b) and Wisconsin (c) datasets when the total number of trees varies. The ARI remains relatively constant.

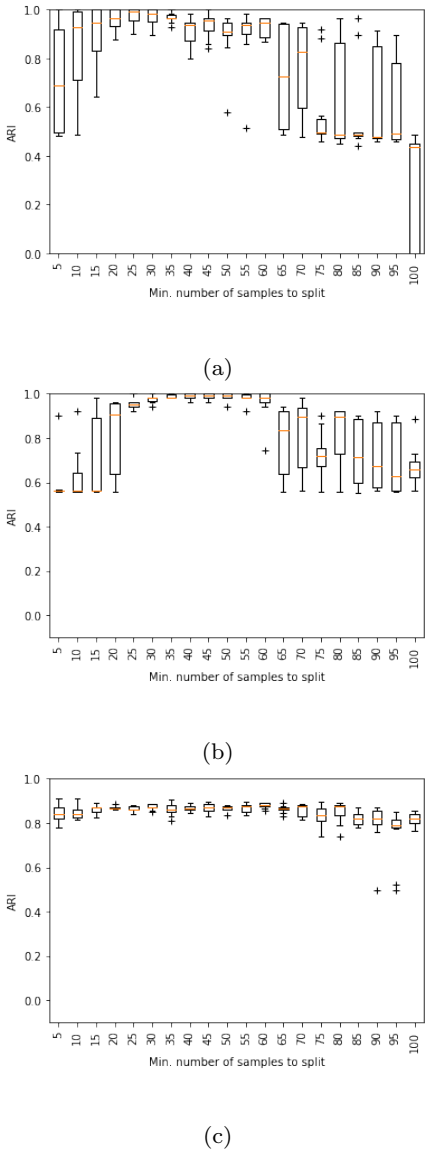

Fig. 2 ARI performing agglomerative clustering using distance matrices computed with UET on Wine (a), Iris (b) and Wisconsin (c) datasets when the min. number of samples to split increases. The values correspond to the percentage of instances in each dataset.

tree growth in order to group similar instances in the same leaves more often. The minimum number of objects to split a node n_{min} can control this growth. This parameter n_{min}, also called the *smoothing strength*, has an impact on the biais and the variance. As stated by Geurts *et al.* [11], the optimal value for this parameter depends on the level of noise in the dataset. They showed in [11] that larger values of n_{min} are needed when ET is applied to noisy data. In UET, the noise is extreme, as the labels are assigned randomly. The same statistical analysis that we performed previously show that there is a significant difference in ARI in the tested datasets according to n_{min} ($p < 0.1$). Values of n_{min} between 20% and 30% of the number of instances seem to give better results. The ARI variations for the three datasets according to n_{min} are presented Figure 2.

These experiments seem to indicate that UET is robust with respect to the number of trees in the ensemble, while being sensitive to the *smoothing strength*. Now that we have an idea of the behaviour of the algorithm with respect to its parameters, we focus on its behaviour on various synthetic and real-world datasets.

3 Empirical evaluation

Our empirical evaluation of UET is divided into 4 parts. In subsection 3.1, we assess if our method can effectively discriminate clusters. Indeed, if that was not the case, UET would not be useful in practice. We also assess its robustness in three scenarios: presence of monotone transformations of variables, correlated variables, and noisy data. In Subsection 3.2, we evaluate the performance of UET on numerical, categorical and heterogeneous datasets. We then compare clusterings using UET as a base similarity measure and clusterings obtained in the litterature in Subsection 3.3. Finally, we compare clusterings obtained with UET with clustering using the euclidean distance in Subsection 3.4. This step ensures that the method we propose here is competitive with a naive, more popular measure.

3.1 Assessment of some characteristics of UET

All the experiments presented here are based on the comparison of the mean intracluster similarities and the mean intercluster similarities, as well as their differences, taking values in the interval $[0, 1]$. These metrics enable a comparison that is agnostic to a subsequent clustering method.

The mean difference is computed as follows. First, the arithmetic mean of the pairwise similarities between all samples having the same label is computed, corresponding to the mean intracluster similarity μ_{intra}. Then the same process is done for samples with a different label, giving the mean intercluster similarity μ_{inter}. We finally compute the difference $\Delta = |\mu_{intra} - \mu_{inter}|$. In our experiments, this difference Δ is computed 20 times. In the following section, $\bar{\Delta}$ denotes the mean of differences between runs, and σ its standard deviation.

3.1.1 Ability of UET to discriminate clusters

Three datasets were generated for this task. Two datasets composed of 1000 samples each have no cluster structure, and differ only in the number of features, 4 and 50. The columns are generated by a sampling from a normal distribution $\mathcal{N}(0, 1)$. These datasets are referred to as *NoC4* and *NoC50*, respectively. We generated another dataset *C4*, where the first 500 lines are drawn from a uniform distribution $\mathcal{U}(0, 0.5)$ for one column and $\mathcal{U}(1, 2)$ for the second column. The other 500 lines are drawn from $\mathcal{U}(0.5, 1)$ and $\mathcal{U}(0, 1)$. We then added two columns, defined as a discretization of the previously generated columns. These columns are categorical, and enable us to test our method on an heterogeneous dataset. The results are presented in Table 3.1.1.

Dataset	$\bar{\Delta}$	σ
NoC4	0.00042	0.00003
NoC50	0.00007	0.00003
C4	0.68417	0.00341

Table 3 Mean difference between intercluster and intracluster similarities in different settings, on synthetic datasets.

In *NoC50*, $\bar{\Delta} \approx 0$, whereas we obtained a value of $\bar{\Delta}$ much higher and closer to 1 in the dataset with a clear cluster structure. This result is not a formal proof but demonstrates that UET is able to clearly discriminate clusters in the data when they exist and to return no significant differences in similarities when no cluster structure is present.

3.1.2 Invariance under monotone transformations

One interest of the proposed method is the invariance to monotone transformations of individual variables. Indeed, as stated by J.H Friedman in [21], this feature provides a resistance to the presence of outliers in the data, as well as any change in the measurement scales of the variables. Here, we assess this property on two synthetic datasets generated by the *make_blobs* and *make_moons* methods of the library *Scikit-learn* [22]. The first one, that we will refer to as **blob500**, contains 500 samples, 5 features and 3 clusters. The second one describes moon-shaped clusters consisting of 500 samples, described by two features, that we will refer to as **moon500**.

After a first computation of $\bar{\Delta}$ on the original data, we iteratively multiply each column of the dataset by a scalar drawn from $\mathcal{U}(2, 100)$, and recompute $\bar{\Delta}$. Results for the **blob500** and **moon500** datasets are presented Table 3.1.2. We observe that UET is effectively robust with respect to monotone transformation of one or many variables, the observed variations being due to the stochastic nature of the method. These results were expected as the split is insensitive to changes if every value in the column is shifted the same way.

Operation	Number of variables	$\bar{\Delta}$	σ
Multiplication	0	0.2981	0.0044
Multiplication	1	0.2991	0.0029
Multiplication	2	0.2992	0.0036
Addition	0	0.2987	0.0037
Addition	1	0.2976	0.0045
Addition	2	0.2970	0.0035

Table 4 Influence of a multiplication or addition by a scalar on $\bar{\Delta}$ for moon500 dataset

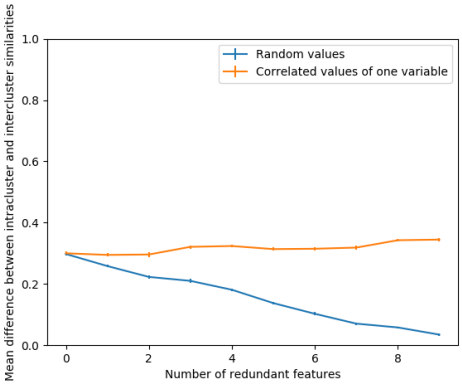

Fig. 3 Change of difference between mean intracluster and mean intercluster similarities when (i) changing features by linear combinations of other features and (ii) changing features by random values. The x axis represents the number of features modified by the procedure.

Operation	Number of variables	$\bar{\Delta}$	σ
Multiplication	0	0.3283	0.0072
Multiplication	1	0.3297	0.0060
Multiplication	2	0.3285	0.0067
Addition	0	0.3250	0.0053
Addition	1	0.3296	0.0046
Addition	0	0.3267	0.0059

Table 5 Influence of a multiplication or addition by a scalar on $\bar{\Delta}$ for blob500 dataset

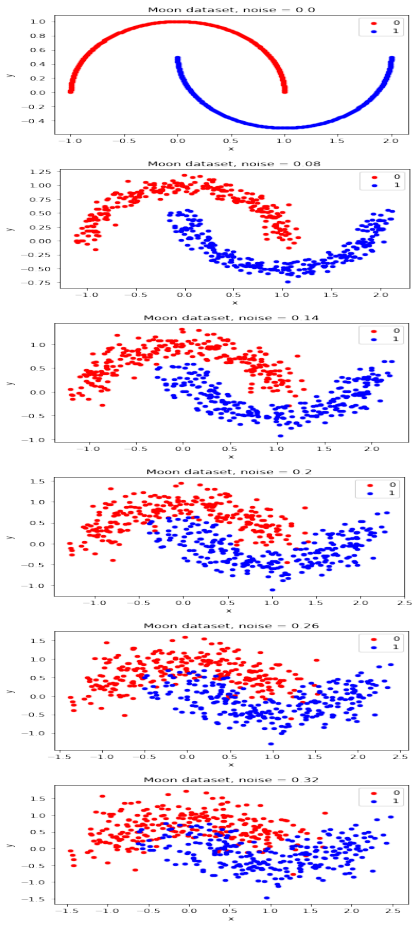

Fig. 4 The figure shows a representation of some of the moon-shaped clusters used for evaluation, with their change with respect to noise. When the standard deviation of the Gaussian noise is greater than 0.20, the boundaries between both clusters begin to merge.

3.1.3 Robustness with respect to correlated variables

These experiments were conducted on the **blob500** dataset, with 10 features instead of 5. We replaced each column of the dataset in an iterative fashion by a random linear combination of another column, and computed $\bar{\Delta}$ and σ. We also ran the same experiment replacing each column by random values. Figure 3 summarizes the results of the experiments. We notice that the presence of linearly correlated variables have little to no effect on the overall $\bar{\Delta}$. This is an interesting feature, as it is not necessary to perform feature selection prior to the computation, hence reducing the number of preprocessing tasks needed.

3.1.4 Robustness with respect to noise

As real-world datasets are often noisy, it is important for any learning method to be robust to noise. To evaluate how the proposed method behaves with noisy data, we first generated 25 **moon500** datasets. Each dataset is generated using the same seed. A Gaussian noise was added, incrementing the value of its standard variation from 0 (*i.e*, without noise) to 0.48. Figure 4 shows some of the moon-shaped datasets used in our experiments.

We also assessed the robustness with respect to noise on two other datasets: **blob500**, a synthetic one, and *Iris* [17], a real-world dataset. As no method to generate noise for these data exist in *Scikit-learn*, we manually added noise. We did so by adding to each value x of the dataset a value drawn from $\mathcal{U}(-st \times x, st \times x)$, st being the strength of the noise.

The results we obtained are interesting. Indeed, while we do have an expected decrease in $\bar{\Delta}$, the slope remains acceptable. Moreover, in the case of the moon-shaped clusters, even if for values of the standard deviation of the Gaussian noise greater than 0.20 the boundaries between both clusters begin to merge, we notice that UET is still able to discriminate between the two clusters.

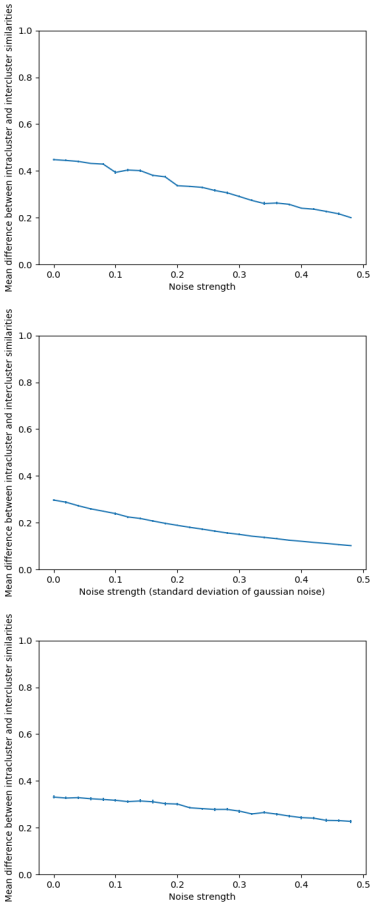

Fig. 5 Evolution of the mean difference between intracluster and intercluster similarity when the noise is increased in three datasets (from left to right : **blob500**, **moon500** and *Iris*.)

The characteristics we assessed here are interesting, as they reduce the number of preprocessing steps that may be needed for some datasets. In [23], Hastie *et al.* described an "off-the-shelf" method as:

[...] one that can be directly applied to the data without requiring a great deal of time- consuming data preprocessing or careful tuning of the learning procedure

Among tree-based methods for similarity computation, UET is a good candidate for an "off-the-shelf" method for clustering purposes.

3.2 Performance of UET on numerical, categorical and heterogeneous data

Another interesting aspect of UET is that it can be applied on different types of data, namely, numerical, categorical or heterogeneous. Here, we focus on these three different settings, and assess whether UET is effectively able to separate clusters.

3.2.1 Numerical datasets

We first assessed UET on purely numerical datasets, described in Table 3.2.1. $S1_n$, $S2_n$ and $S3_n$ are 3 generated datasets, where the values of each column for each cluster are drawn from a different distribution. Iris and Wisconsin dataset were downloaded from the UCI website [1]. The results show that, in these settings, the method effectively returns a significant difference of similarities between samples belonging to the same cluster and other samples.

3.2.2 Categorical datasets

We applied UET on categorical-only datasets. $S1_c$ and $S2_c$ are generated datasets, where the values for each variable and for each cluster are drawn from different multinomial distributions and has 3 modalities. We also applied our method on Soybean dataset, downloaded from the UCI website. These three datasets are described in Table 3.2.2. The results (Table 3.2.2) shows that UET effectively discriminates clusters in purely categorical settings. Indeed, for $S1_c$, $S2_c$ and Soybean, the $\bar{\Delta}$ returned by UET are significantly greater than 0.

3.2.3 Heterogeneous datasets

We finally used UET on heterogeneous on both generated and real-world data to assess its results. The datasets parameters and the results are presented Table 3.2.3. The synthetic datasets $S1_h$, $S2_h$ and $S3_h$ are generated in a similar fashion than the synthetic continuous and categorical. Both real-world datasets, Credit and CMC, were downloaded from the UCI website. We notice that the discriminative power of UET in the heterogeneous case remains similar to the homogeneous one, in most cases. However, for the CMC dataset, $\bar{\Delta}$ is close to 0, meaning that almost no difference is found between instances that belong to different classes. This may be due to the fact that there is no inherent cluster structure in this dataset. Indeed, both real-world datasets used in this settings are primarily used for classification tasks. The labels may not correspond perfectly to some *natural clustering* of the data.

[1] https://archive.ics.uci.edu/ml/index.php

Name	# samples	# variables	# Classes	$\bar{\Delta}$	σ
$S1_n$	500	2	2	0.4685	0.0015
$S2_n$	450	2	3	0.5290	0.0007
$S3_n$	500	4	2	0.4873	0.0082
Iris	150	4	3	0.4312	0.0056
Wisconsin	699	9	2	0.2259	0.0048

Table 6 Mean differences between intracluster and intercluster similarities computed with UET, denoted by $\bar{\Delta}$, while σ denotes the standard deviation.

Name	# samples	# variables	# Classes	$\bar{\Delta}$	σ
$S1_c$	500	2	2	0.3878	0.0056
$S2_c$	500	4	2	0.3387	0.006
Soybean	266	35	19	0.2370	0.0062

Table 7 Mean differences between intracluster and intercluster similarities computed with UET, denoted by $\bar{\Delta}$, while σ denotes the standard deviation.

Name	# samples	# Continuous	#Categorical	# Classes	$\bar{\Delta}$	σ
$S1_h$	500	2	2	2	0.3009	0.0061
$S2_h$	500	2	4	2	0.3210	0.0058
$S3_h$	500	4	4	2	0.3557	0.0063
Credit	690	7	6	2	0.0648	0.0026
CMC	1473	5	4	3	0.0118	0.0003

Table 8 Mean differences between intracluster and intercluster similarities computed with UET, denoted by $\bar{\Delta}$, while σ denotes the standard deviation.

3.3 Comparative evaluation with results from the literature

In the previous sections, we compared empirically the distances between intra- and inter-cluster similarities. Although this metric is appealing, as it reduces the bias associated with a specific choice of clustering method, it is interesting to compare some results on real-life clusterings with the litterature, obtained using common similarity measures.

We adopted the following protocol. For each dataset, UET was run 10 times, and the similarity matrices were averaged. The thus obtained matrix was then transformed into a distance matrix using the equation $DIS_{ij} = \sqrt{1 - SIM_{ij}}$, and an agglomerative clustering with the relevant number of clusters was performed. The quality of the clustering was then evaluated with respect to normalized mutual information (NMI). Equation (2) shows how this metric is computed, where X is the class labels, Y the clusters, $H()$ denotes the Shannon's entropy, and $I(X,Y)$ the mutual information, where $I(X,Y) = H(X) - H(X|Y)$. NMI values lie in the range $[0,1]$. Here, the values presented are multiplied by 100.

$$NMI(X,Y) = \frac{2 \times I(X,Y)}{H(X) + H(Y)} \quad (2)$$

The clustering is run 20 times, and we provide the mean and standard deviation of the NMI. This evaluation was performed using *scikit-learn* [22] and our implementation of UET. This implementation will be available upon request.

The ten datasets used in this section are available on the UCI website [2] and presented Table 3.3.

In [24], NMI obtained by running k-means 20 times and averaging the results are provided for each dataset. We compared our results to the ones obtained without feature selection, as none has been performed in our setting. As the results from [24] are used only for a clustering performance comparison, we did not perform any time comparison. Moreover, the method presented in their paper being a feature selection one, a fair comparison between execution time is not trivial. The results are presented Table 3.3. They show that NMI scores obtained using UET are competitive in most cases. It is noteworthy that in some cases, UET without feature

[2] https://archive.ics.uci.edu/ml/index.php

Dataset	# samples	# features	# labels
Iris	150	4	3
Wine	178	13	3
Wisconsin	699	9	2
Lung	32	56	3
Breast tissue	106	9	6
Isolet	1559	617	26
Pima	768	8	2
Parkinson	195	22	2
Ionosphere	351	34	2
Segmentation	2310	19	7

Table 9 Datasets used for benchmarking

Dataset	UET - NMI	Literature - NMI
Wisconsin	**78.33** ± 3.25	73.61 ± 0.00
Lung	**29.98** ± 6.17	22.51 ± 5.58
Breast tissue	**74.48** ± 2.92	51.18 ± 1.38
Isolet	61.22 ± 1.47	**69.83** ± 1.74
Parkinson	**25.50** ± 6.14	**23.35** ± 0.19
Ionosphere	**13.47** ± 1.11	**12.62** ± 2.37
Segmentation	**69.62** ± 2.14	60.73 ± 1.71

Table 10 Comparative evaluation with the results from [24]. Best obtained values are indicated in boldface. In case of a tie, both values are in boldface.

selection gives better results than the ones obtained by [24] after feature selection. For instance, this is the case for *Breast tissue* dataset.

3.3.1 Comparison with URF

To compare UET and URF, we used the R implementation provided by Shi and Horvath [3], and compared the ARI obtained after running the partitioning around medoids (PAM) algorithm on the distance matrices obtained by both methods. 2000 trees and 100 forests are used for URF, with a value of $m_{try} = \lfloor \sqrt{n_{features}} \rfloor$ [4]. We set UET parameters to $n_{trees} = 200$ and $n_{min} = \lfloor \frac{n_{samples}}{3} \rfloor$, and averaged the similarity matrices of 20 runs. These experiments were run on a computer with an Intel i7-6600U (2.6 Ghz) and 16 Go of 2133 MHz DDR4 RAM.

We compared both ARI and time (in seconds) for each method. The results are presented Table 3.3.1. UET outperforms URF time-wise, while giving similar or better clusterings. Regarding the Isolet dataset, we had to stop URF's computation as we weren't able to obtain results in an acceptable amount of time on our machine. However, we performed the computation on a more powerful machine, and were able to obtain an ARI of 28.39.

[3] `https://labs.genetics.ucla.edu/horvath/RFclustering/RFclustering.htm`

[4] m_{try} is the number of variables used at each node when a tree is grown in RF.

Dataset	UET (ARI - Time)	URF (ARI - Time)
Wisconsin	**87.13 - 128.42 s**	82.92 - 968.71 s
Lung	**23.24 - 5.23 s**	6.52 - 86.93 s
Breast tissue	**58.85 - 9.15 s**	39.05 - 99.40 s
Isolet	**28.04 - 692.82 s**	* - * s
Parkinson	**25.21 - 16.27 s**	12.68 - 279.30 s
Ionosphere	6.04 - **39.13 s**	**7.28** - 727.30 s

Table 11 Comparative evaluation between URF and UET. Best obtained values are indicated in boldface. In case of a tie, both values are in boldface.

Dataset	UET - NMI	Euclidean - NMI
Wisconsin	**79.32** ± 2.12	66.03 ± 0.00
Lung	**28.64** ± 4.90	**28.20** ± 0.00
Iris	**98.21** ± 1.5	80.58 ± 0.00
Wine	**95.01** ± 4.74	41.58 ± 0.00
Parkinson	**30.37** ± 3.90	0.00 ± 0.19
Soybean	**85.02** ± 1.58	71.86 ± 0.00
Pima	**2.80** ± 0.90	0.00 ± 0.00

Table 12 Comparative evaluation between URF and euclidean distance. Best obtained values are indicated in boldface. In case of a tie, both values are in boldface.

3.4 Comparison with euclidean distance

We finally compared the clusterings obtained using UET with clusterings using other similarity or distance metrics. The procedure was the following: for continuous datasets, we performed HAC using the euclidean distances. For categorical ones, we first used One-Hot encoding to transform the features, and then applied HAC using the euclidean distances. We finally compared the NMI obtained using these methods with the NMI obtained using UET. The results are presented Table 3.4.

These datasets are either small or low-dimensional ones. As it is interesting to assess how the method behaves on larger datasets, we performed the same experiments on two other high-dimensional datasets presented Table 3.4. For both datasets, the clustering obtained using UET outperforms the clustering obtained using euclidean distance ($p < 0.01$). Yet, the advantage of using euclidean distance is its non stochasticity. Indeed, it is not necessary to repeat the experiment as

Dataset	# samples	# features	# labels	UET - NMI	Euclidean - NMI
Olivetti faces	400	4096	40	**79.32** ± 2.12	73.69 ± 0.00
Digits	1797	64	10	**94.54** ± 0.99	71.61 ± 0.00

Table 13 NMI obtained on images datasets. Best obtained values are indicated in boldface. In case of a tie, both values are in boldface.

the distance will remain the same in each run, hence the standard deviation of 0.

4 Conclusion and perspectives

In this work, we presented a novel method to compute pairwise similarities using stochastic decision trees. This approach extends the unsupervised random forest method, by using extremely randomized trees as a base estimator. In URF, the generation of synthetic instances was needed and performed by two different approaches, *AddCl1* or *AddCl2*. With UET, the generation of instances is no longer necessary: by setting the number of attributes to be drawn at each split $K = 1$, extremely randomized trees can be made independent of the labels. We therefore present a way to bypass the need for instance and label generation, which results in a significant reduction in running time.

UET also enables the computation of pairwise similarities between samples in both homogeneous and heterogeneous datasets, and has some interesting properties that we assessed empirically in this work. Some of these properties are appealing since they drastically reduce the preprocessing burden. For instance, the invariance with respect to monotone transformations of individual or multiple variables remove the need for scaling that exists for euclidean distance-based methods such as *k-means*. Moreover, in practice, questions regarding the management of highly correlated attributes arise. The robustness of the method in cases where variables are redundant indicates a potential solution in these cases.

The performance evaluation of our method showed that essentially one parameter influenced the results, the *smoothing strength* n_{min}. This is explained by the fact that higher values of n_{min} give better results in the presence of noise. In our case, the data is highly noisy, as the splits are randomly performed without any label consideration. We found that a value of $\frac{n_{samples}}{4} \leq n_{min} \leq \frac{n_{samples}}{3}$ gives good clusterings. Other parameters, such as the number of trees per forest n_{trees} did not impact much the results of the procedure for values of $n_{trees} > 50$, while increasing the time to perform the procedure.

We compared the quality of clusterings between our method and (i) results found in the literature and (ii) results obtained by URF on multiple datasets. The quality was measured by normalized mutual information or adjusted rand index, according to the metric available in the literature. This empirical evaluation gave good results, with overall similar or better NMI and ARI. The advantages of our method over URF are twofold. First, the generation of synthetic data is no longer necessary. Second, the method is 1.5 to more than 10 times faster than URF.

Although the empirical evaluation of UET provided good overall results, there are still some issues that are currently being considered. Specifically, for some datasets, we obtained some unexpected results since the method was unable to effectively discriminate clusters. In such cases, some fine-tuning of the smoothing strength may improve the results. Indeed, high values of n_{min} might be unsuitable for some datasets, as only a few variables are drawn overall in the trees growth. We also observed a discrepancy in some heterogeneous datasets. These results may be caused by some of their characteristics that are still being investigated.

Complexity issues are currently being considered. Indeed, the size of the similarity matrix returned by UET can become cumbersome for datasets with a large number of instances ($> 10,000$, typically). Moreover, the computational complexity of UET may be an issue for these datasets. We already made some improvements taking advantage of the sparsity of the matrices and rewriting our implementation. Moreover, UET being an ensemble method, all computations can easily be distributed across a cluster of machines.

Finally, KM Ting *et al.* [25] recently proposed a similar approach to compute a mass-based dissimilarity between instances, based on isolation forests [26]. While their approach is similar to the one we present here, it differs on some key points, such as the fact that self-similarities are not constant in mass-based dissimilarity, as $m_e(x, x) \in [0, 1]$ depends on the distribution of the data. This feature is interesting and provides good results in some cases, must notably for density-based clustering where clusters are of varying densities, but makes it hard to trivially transform dissimilarities to similarities. Although they differ on some specific features, the ideas behind both methods are very similar

and it would be interesting to compare them extensively.

Overall, UET can be a good candidate method in exploratory data analysis where handling heterogeneity and preprocessing tasks often reveal to be cumbersome.

Acknowledgements

Kevin Dalleau's PhD is funded by the RHU FIGHT-HF (ANR-15-RHUS-0004) and the Region Grand Est (France).

On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. M. M. Deza and E. Deza. Encyclopedia of distances. In *Encyclopedia of Distances*, pages 1–583. Springer, 2009.
2. T. Shi and S. Horvath. Unsupervised learning with random forest predictors. *Journal of Computational and Graphical Statistics*, 15(1):118–138, 2006.
3. L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
4. B. Percha, Y. Garten, and R. B. Altman. Discovery and explanation of drug-drug interactions via text mining. In *Pacific Symposium on Biocomputing*, pages 410–421, 2012.
5. M. Pal. Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1):217–222, 2005.
6. J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.
7. H. L. Kim, D. Seligson, X. Liu, N. Janzen, M.H. Bui, H. Yu, T. Shi, A. S. Belldegrun, S. Horvath, and R.A. Figlin. Using tumor markers to predict the survival of patients with metastatic renal cell carcinoma. *The Journal of urology*, 173(5):1496–1501, 2005.
8. M. C. Abba, H. Sun, K. A. Hawkins, J. A. Drake, Y. Hu, M. I. Nunez, S. Gaddis, T. Shi, S. Horvath, and A. Sahin, *et al.* Breast cancer molecular signatures as determined by sage: correlation with lymph node status. *Molecular Cancer Research*, 5(9):881–890, 2007.
9. S. I. Rennard, N. Locantore, B. Delafont, R. Tal-Singer, E. K. Silverman, J. Vestbo, B. E. Miller, P. Bakke, B. Celli, and P.M. Calverley, *et al.* Identification of five chronic obstructive pulmonary disease subgroups with different prognoses in the eclipse cohort using cluster analysis. *Annals of the American Thoracic Society*, 12(3):303–312, 2015.
10. K.Y. Peerbhay, O. Mutanga, and R. Ismail. Random forests unsupervised classification: The detection and mapping of solanum mauritianum infestations in plantation forestry using hyperspectral data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(6):3107–3122, 2015.
11. P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.
12. Vineet K Raghu, Joseph D Ramsey, Alison Morris, Dimitrios V Manatakis, Peter Sprites, Panos K Chrysanthis, Clark Glymour, and Panayiotis V Benos. Comparison of strategies for scalable causal discovery of latent variable models from mixed data. *International Journal of Data Science and Analytics*, pages 1–13, 2018.
13. Michail Tsagris, Giorgos Borboudakis, Vincenzo Lagani, and Ioannis Tsamardinos. Constraint-based causal discovery with mixed data. *International Journal of Data Science and Analytics*, pages 1–12, 2018.
14. L. Kaufman and P. J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.
15. W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.
16. L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.
17. R.A. Fisher and M. Marshall. Iris data set. *RA Fisher, UC Irvine Machine Learning Repository*, 1936.
18. M. Forina, *et al.* An extendible package for data exploration, classification and correlation. *Institute of Pharmaceutical and Food Analysis and Technologies*, 16147, 1991.
19. O.L. Mangasarian and W.H. Wolberg. Cancer diagnosis via linear programming. *Computer Sciences Department, University of Wisconsin-Madison*, 1990.
20. W.H. Kruskal and W.A. Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260):583–621, 1952.
21. Jerome H Friedman. Recent advances in predictive (machine) learning. *Journal of classification*, 23(2):175–197, 2006.
22. F. Pedregosa, *et al.* Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
23. Hastie Trevor, Tibshirani Robert, and Friedman JH. The elements of statistical learning: data mining, inference, and prediction, 2009.
24. H. Elghazel and A. Aussem. Feature selection for unsupervised learning using random cluster ensembles. In *2010 IEEE 10th International Conference on Data Mining (ICDM)*, pages 168–175. IEEE, 2010.
25. Kai Ming Ting, Ye Zhu, Mark Carman, Yue Zhu, Takashi Washio, and Zhi-Hua Zhou. Lowest probability mass

neighbour algorithms: Relaxing the metric constraint in distance-based neighbourhood algorithms. *Machine Learning*, 2018.

26. Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.